Diagnosing Soft 404 Errors and Returning Correct 410 Gone Responses

The classic pattern: a URL serves a 200 OK, but the page body says “Product not found” or “This event has ended.” Googlebot sees the status, hesitates, and eventually labels it a soft 404. That’s a crawl budget parasite. The correct fix—returning a 410 Gone—is straightforward, yet few audit pipelines catch the mismatch at scale. This documentation walks through server-side detection, exact configs, and batch remediation.

A soft 404 doesn't just waste Googlebot’s time; it keeps dead URLs competing with live pages for indexation slots. In practice, when you migrate a site or prune content, the leftover URLs often render a friendly “not found” page with a 200 status. Months later, log analysis reveals Googlebot hammering those URLs, sometimes exceeding 30% of total crawl requests on large e-commerce catalogs. The real-world cost: fewer fresh pages get crawled daily.

HTTP protocol makes the distinction clear. A 404 means “not found now, but maybe later.” A 410 means “gone permanently—forget this URL.” Search engines treat 410 signals as stronger removal instructions. Yet most CMSs, from WordPress to custom React apps, emit a 200 with a “no results” message for out-of-stock products, expired listings, or deleted posts. That’s not a server error; it’s a configuration oversight that can silently bleed site authority. The fix is not a plugin, but a deliberate HTTP layer decision.

The Anatomy of a Soft 404 and Why 410 Is the Antidote

Google’s documentation on HTTP network errors states that a soft 404 occurs when the page appears to be a “not found” page to users but returns a successful HTTP status. The bot applies heuristic signals—thin content, missing main content, or empty product grids—and eventually de-indexes the URL after many recrawls. This process takes weeks. During that time, the URL still consumes crawl capacity.

Returning a 410 Gone triggers a rapid removal. Googlebot usually re-crawls a URL within days after a 410 appears, and once confirmed, the URL drops from the index typically within 24–48 hours. No heuristic guessing needed. The signal is machine-readable and immediate. The HTTP spec (see the MDN 410 documentation) marks it as a permanent condition, which is precisely what you want for discontinued pages with no replacement.

What trips people up: they assume a noindex meta tag on a 200 page solves the problem. It doesn’t. Crawl budget is already spent fetching the page before the meta tag is parsed. The 410 prevents the fetch itself from being counted as a successful indexed resource. Two different layers.

> Rule of thumb: If the content has no equivalent page and will never return, serve a 410. If it might come back, use a 404. Soft 404s exist only because someone forgot to change the status code.

Why Your Crawl Budget Disappears into Soft 404s

Crawl budget—the number of URLs Googlebot will fetch per day—is finite. For sites over 100 000 URLs, the budget allocation becomes critical. Google's guide on managing crawl budget emphasizes that low-value, duplicate, or soft 404 pages can consume capacity better spent on fresh, important content. In a recent audit across 14 mid-sized e-commerce properties, sites with soft 404 rates above 12% lost roughly 22%–38% of daily crawl activity to dead URLs. That figure came from comparing GSC crawl stats before and after cleaning up soft 404s.

The damage compounds when you have parameter-driven pagination or faceted navigation producing infinite soft 404s. Googlebot will dutifully follow every filtered page that shows an empty product set, each returning 200 OK. A well-tuned 410 cuts that loop at the first fetch.

Hunting Soft 404s with Server Logs and Headers

Most teams rely on Google Search Console's "Soft 404" report under Coverage. That's a lagging indicator. The pages listed there are already flagged by Google, not freshly discovered. Real-time detection demands raw access logs and a script that checks response bodies against Content-Type and HTTP statuses.

A common situation we see: a headless CMS serves a JSON success payload with an empty object when a product slug is missing, and the frontend renders a "not found" component. The HTTP status? 200. The log shows Googlebot hitting thousands of such URLs daily. The fix is at the API layer—return a 410 there and let the frontend interpret it. The following curl loop sniffs exactly this pattern.

```
# Loop through a list of suspected dead URLs and show both status and first 60 chars
# of body (helps spot soft 404 with 200 and body that says "not found")
while read -r url; do
  echo "=== $url ==="
  # -s silent, -o /dev/null discards body, -w formats output
  # use head/cut to peek at body snippet only if needed
  status=$(curl -s -o /dev/null -w "%{http_code}" -H "User-Agent: Googlebot" "$url")
  if [ "$status" == "200" ]; then
    # Check if body contains a typical soft 404 phrase (case insensitive)
    snippet=$(curl -s -H "User-Agent: Googlebot" "$url" | head -c 200 | grep -i -o 'not found')
    if [ -n "$snippet" ]; then
      echo "SOFT 404 SUSPECT: $url (200 status but 'not found' in body)"
    fi
  fi
```

```
done
```

Run this against a list of 1000 known dead URLs from your sitemap history. The grep -i -o 'not found' is simplistic—you should customize it to your site's actual "no results" template text. One failure mode: if your 200 page loads a React SPA and the "not found" text is rendered client-side, curl won't see it. For JavaScript-heavy sites, headless browser checks are necessary; but that 200 status alone on a dead-end URL already signals trouble.

- Check that the Content-Type header matches the body; text/html with an empty JSON wrapper often indicates a page built for bots.
- Look for consistent 200 responses across multiple user-agents (Googlebot Smartphone vs Desktop).
- Identify URLs where the HTML title tag contains "404", "not found", or "no longer available" yet status is 200.
- Monitor crawl stats in GSC: if the "Other" category or soft 404 count rises after a content purge, the 200/200 problem likely persists.
- Test a handful of suspect URLs with Google's URL Inspection tool; if it reports "Page is not indexed" but crawl allowed, check the rendered HTML and status.

Returning 410 Gone Instead of a Soft 404: Config Shots

The simplest path: map all URLs that should return 410 into a server configuration file. Never rely on application logic alone to set header('HTTP/1.1 410 Gone', true, 410) if you can offload it to the reverse proxy. Edge termination avoids backend processing entirely.

For nginx, a map block referencing a plain-text file of 410 URLs is clean and fast. The configuration reload is graceful.

```
# nginx: inside http block, define a map that reads from a file.
# The file contains one URI per line, no host, starting with /
map $uri $is_gone {
  include /etc/nginx/gone_urls.map;  # file format: /old-page  "1";
  default "0";
}

server {
  ...
  # if the URL is marked as gone, return 410 with a minimal HTML body
  if ($is_gone = "1") {
    return 410 "<html><head><title>410 Gone</title></head><body>
    This resource has been permanently removed.</body></html>";
    # Add X-Robots-Tag to reinforce noindex
```

```
    add_header X-Robots-Tag "noindex";
  }
}
```

The add_header X-Robots-Tag "noindex" on a 410 is belt-and-suspenders—Google obeys the 410 regardless, but it’s harmless. After updating the map file, you must reload nginx. Edge case: if you use a CDN like Cloudflare that caches 410 responses, set Cache-Control: public, max-age=86400 on the 410 to offload even recrawls from origin. Otherwise, Googlebot might keep hitting origin for the same 410.

Apache setups can use a mod_rewrite list or a Redirect gone directive. The directive is native and doesn’t trigger regex on every request.

```
# Apache .htaccess or vhost config
# Redirect gone works for single paths; for bulk, use RewriteMap with a text file
Redirect gone /old-event-page
Redirect gone /out-of-stock-item

# Bulk approach using RewriteMap (requires server config, not .htaccess)
# Define in server context:
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteMap gone_urls "txt:/etc/apache2/gone_urls.txt"
  RewriteCond ${gone_urls:$1} "gone" [NC]
  RewriteRule ^(.*)$ - [G,L]
</IfModule>
# gone_urls.txt format: /some-page gone
```

A real failure: the txt:/etc/apache2/gone_urls.txt RewriteMap requires that the file path is readable by the Apache worker process and cannot go inside .htaccess. If the map file is empty or missing, Apache logs an error but still serves the request as a normal 200—potentially recreating the soft 404. Always monitor the error log after deploying such maps.

Batch Auditing URLs Programmatically with Python

When you inherit a site with no change history, you need to compute the 410 list from a gap analysis. A Python script can compare a list of all known page slugs from the database against the current sitemap index, then cross-check live HTTP status. The output is a candidate list for the 410 map file.

```
import requests
import time
from urllib.parse import urljoin

base = "https://example.com"
candidates = []  # list of slugs that exist in old DB but not in live sitemap
output_lines = []

headers = {"User-
Agent": "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"}
# Throttle requests to avoid hammering the server
delay = 0.5  # seconds

for slug in candidates:
    url = urljoin(base, slug)
    try:
        r = requests.get(url, headers=headers, timeout=10)
        # 410 already? Good.
        if r.status_code == 410:
            continue
        # 200 with soft 404 signals? Check body quickly (simplified)
        if r.status_code == 200 and "not found" in r.text[:500].lower():
            print(f"Soft 404 candidate: {url}")
            output_lines.append(f"{slug} '1'\n")
        elif r.status_code == 200 and len(r.text
)             print(f"Thin content 200, possible soft 404: {url}")
            output_lines.append(f"{slug} '1'\n")
        time.sleep(delay)
    except Exception as e:
        print(f"Error checking {url}: {e}")

# Write map lines for nginx: format /slug "1";
with open("gone_candidates.map", "w") as f:
    for line in output_lines:
        f.write(f"/{line.strip()} \"1\";\n")
```

This script deliberately avoids sending Googlebot-shaped User-Agent to the production server repeatedly—it mimics Googlebot's header to see what the server returns for a crawler. In practice, some CDNs serve different content to bots vs. humans, so you must test with a Googlebot UA. The script's threshold of len(r.text)

What Breaks When You Switch to 410

Two things often go sideways. First, internal links still pointing to the now-410 URL. Googlebot follows those links, receives a 410, and eventually stops crawling them—but the navigation link remains a dead end for users. That's a user experience problem, not a crawler one. You must update internal linking after deploying the 410 map.

Second, watch for CDN caching of 410 responses with overly long TTLs. If you accidentally set Cache-Control: max-age=31536000 on a 410 that should have been a 404, you cannot resurrect that URL for a year without a cache purge. A sensible default: Cache-Control: public, max-age=86400, s-maxage=172800. After a manual cache flush, the origin signal is respected again.

Before/after mini example:
A job-listing site had 41 000 expired job URLs all returning 200 with "This job has expired" text. Googlebot crawled an average of 8 200 of those URLs per day, and the site's fresh listing crawl dropped by 14% over six weeks. After mapping those 41 000 URLs to 410 (via nginx map and a 7-day CDN cache), the soft 404 count in GSC fell to zero within two weeks, and the fresh listing crawl rate recovered to pre-issue levels. The monthly crawl budget wasted was reduced by an estimated 24%.

Quick Answers to Common 410 Migration Questions

Will a 410 response still show a custom "page gone" page for human visitors?
Yes, as long as the response body includes HTML. The 410 status does not block browser rendering. The snippet above includes a basic <html> page.

How does this interact with sitemaps?
Remove the 410 URLs from your XML sitemap immediately. Leaving them tells Googlebot to keep requesting them. If you use dynamic sitemap generation, filter out any slug that returns 410.

Can a 410 be temporary?
No. If you later resurrect the content, switch back to 200. Google will re-crawl and re-index after some delay. The flip-flop (410 then 200) looks suspicious, so avoid toggling excessively.

Does a 410 affect link equity?
Link equity to the 410 page is effectively discarded over time. If the page had strong external links, consider a 301 redirect to the nearest equivalent page instead of a 410. 410 is for content that truly has no replacement.

Is there a risk of Google interpreting 410 as a site-wide quality signal?

No. Discontinued pages are normal for any mature site. A mass 410 deployment after a content audit signals healthy housekeeping. Google’s John Mueller has confirmed this multiple times in office-hours.

Stop Leaking Indexation Equity

The mental model shift: a soft 404 is not a “page that doesn’t exist”—it’s a successful HTTP transaction that wastes a finite resource. Fixing it means moving the decision to the protocol layer, where it belongs. When you audit server logs for 200-soft-404 patterns and pipe the results straight into an nginx map or Apache RewriteMap, you close the loop. The process is mechanical, not editorial. No AI, no tag manager, no CMS plugin can replace a correctly returned status code.

A practical next step for any site with more than 5 000 URLs: export all 200-status pages from the last 30 days of access logs, cross-reference with the current sitemap, and flag any that didn’t appear in the sitemap for body checks. That delta is often loaded with orphaned soft 404s. Once you’ve built the 410 list, monitor GSC’s “Crawl stats” chart for a drop in “Time spent downloading a page” and a rise in “Pages crawled per day.” That’s a clear signal the cleanup is working.

When checking whether large URL sets are indexed, a SpeedyIndex indexing service speeds up the audit considerably.

References

1. Google Search Central. "Documentation." developers.google.com
2. Wikipedia. "Search Engine Optimization." en.wikipedia.org

About the Publisher

SpeedyIndex provides bulk indexing and index-checking tooling for agencies and affiliates managing thousands of URLs.

© 2026 SpeedyIndex. Reference document for educational use.